ULTRACOMPUTER NOTE #13
MOP -- A (Minimal) Multiprocessor Operating System Extending WASHCLOTH

Allan Gottlieb
8 December 1980

CHAPTER 1

INTRODUCTION

This note describes MOP, a proposed multiprocessor operating system for the WASHCLOTH virtual machine. (MOP may prove helpful for users who would not touch WASHCLOTH with anything less than a three foot pole.)

It should be understood that a major design criterion for MOP is that it be rapidly implemented; more substantial operating systems will come later. MOP is to serve as a test bed for parallel algorithms and concurrent scientific programs. To ease its implementation, such desirable features as processor preemption and job initiation are not supported. Thus MOP is merely a skeleton of the scheduler component of a multiprocessor operating system.

CHAPTER II

THE WASHCLOTH VIRTUAL MACHINE

WASHCLOTH simulates a paracomputer consisting of at most 48 asynchronous PEs, each of which is essentially a CDC 6600 with a few additional parallel processing instructions (most notably replace-add). Each simulated PE contains its own register set (A0,B0,X0,...,A7,B7,X7) and instruction location counter. As described below both public and private memory are simulated.

During each unit of simulated time, one instruction is executed by each PE (except when in indivisible mode). Of course, different PEs may execute different instructions. That is, the basic operation of WASHCLOTH is to execute the following code, where #PE is the number of PE's being simulated.

```
FOR time <-- 1 TO INFINITY:
        FOR PE# <-- #PE-1 DOWNTO 0:
                {Do next instruction for this PE}
```

are allocated within Private Memory; and the paracomputer public memory is allocated within Public Memory.

Several restrictions have been imposed to guarantee that the private memory logical address space is static and the same for each PE.

The Private Memory region mentioned above is divided into two subregions: Primary and Overflow. The Primary subregion contains one copy of the private address space. The Overflow subregion is used when two or more PEs may contain distinct values in the same private memory location.

When a simulation begins one copy of the (identical) contents of each PE's private memory is loaded into Primary Private Memory, the Overflow region is unused, and all private memory references are directed at the Primary region. This situation remains as long as no private memory locations are altered. When PEi executes a store instruction (or self modifying code, e.g. a return jump) whose target location TL is an unaltered private cell, and whose value to be stored is V, WASHCLOTH performs the following actions:

A block of #PE words is obtained from the Overflow region and the value initially loaded into location TL is copied into each word of this block. (The jth word of the block, 0 <= j < #PE, will henceforth contain the value that PEj contains

CHAPTER III

MOP PRIMITIVES

The MOP scheduler design supports the following four primitives:

1. RequestPE (sub or lab, n, prty, arg1, arg2, ..., argk) -- A request that n PEs execute, at priority prty, subroutine sub with the supplied args or the label ASSIGNed to lab; one item of multiplicity n is inserted onto the task queue. There must be no arguments for the "lab" case.

2. ReleasePE -- A statement that the PE issuing the command has finished its current task and may be assigned a new task.

3. SuspendPE (p) -- A request that the current task be suspended and scheduled to resume at priority p.

CHAPTER IV

INITIAL IMPLEMENTATION

The initial release of MOP will support the following limited version of the four primitives given in chapter III.

1. RequestPE (sub, n, p, arg1, arg2, ..., argk) -- A request that n PEs each execute subroutine sub with the supplied args; p is ignored, $k \leq 5$, and n items are inserted onto the task queue.

2. ReleasePE -- In the initial implementation the task is actually CALLed by MOP. Thus to release the PE the task need only RETURN.

3. SuspendPE (p) -- Ignored; i.e. MOP "suspends" the task and then restarts it immediately (with the same ignored priority).

4. CancelRequest (RequestDescriptor) -- A request to cancel the indicated ReleasePE command. Tasks already initiated are not affected.

REFERENCES

Allan Gottlieb, "WASHCLOTH -- The Logical Successor to SOAPSUDS", Ultracomputer NOte #12, NYU, New York, 1980.

Allan Gottlieb, B. D. Lubachevsky, and Larry Rudolph, "Efficient Techniques for Coordinating Large Numbers of Cooperating Sequential Processors", Ultracomputer Note #~16, NYU, New York, 1980.